Unsupervised Feature Learning for low-level Local Image Descriptors

Christian Osendorfer, Justin Bayer, Sebastian Urban, Patrick van der Smagt
Technische Universität München
{osendorf, bayerj, surban, smagt}@in.tum.de

Abstract

Unsupervised feature learning has shown impressive results for a wide range of input modalities, in particular for object classification tasks in computer vision. Using a large amount of unlabeled data, unsupervised feature learning methods are utilized to construct high-level representations that are discriminative enough for subsequently trained supervised classification algorithms. However, it has never been *quantitatively* investigated yet how well unsupervised learning methods can find *low-level representations* for image patches without any additional supervision. In this paper we examine the performance of pure unsupervised methods on a low-level correspondence task, a problem that is central to many Computer Vision applications. We find that a special type of Restricted Boltzmann Machines (RBMs) performs comparably to hand-crafted descriptors. Additionally, a simple binarization scheme produces compact representations that perform better than several state-of-the-art descriptors.

1 Introduction

In this paper we tackle a recent computer vision dataset [2] from the viewpoint of unsupervised feature learning. Why yet another dataset? There are already enough datasets that serve well for evaluating feature learning algorithms. In particular for feature learning from image data, several well-established benchmarks exist: Caltech-101 [10], CIFAR-10 [19], NORB [23], to name only a few. Notably, these benchmarks are all object classification tasks. Unsupervised learning algorithms are evaluated by considering how well a subsequent supervised classification algorithm performs on high-level features that are found by aggregating the learned low-level representations [8]. We think that mingling these steps makes it difficult to assess the quality of the unsupervised algorithms. A more direct way is needed to evaluate these methods, preferably where a subsequent supervised learning step is completely optional.

We are not only at odds with the methodology of evaluating unsupervised learning algorithms. General object classification tasks are always based on orientation- and scale-rectified pictures with objects or themes firmly centered in the middle. We are looking for a dataset where it is possible to show that unsupervised feature learning is beneficial to the wide range of Computer Vision tasks beyond object classification, like tracking, stereo vision, panoramic stitching or structure from motion. One might argue, that object classification acts as a good proxy for all these other tasks but this hypothesis has not shown to be correct either theoretically or through empirical evidence. Instead, we chose the most general and direct task that can be be applied to *low-level representations*: matching these representations, i.e. determining if two data samples are similar given their learned representation.

Matching image descriptors is a central problem in Computer Vision, so hand-crafted descriptors are always evaluated with respect to this task [28]. Given a dataset of labeled correspondences, *supervised* learning approaches will find representations *and* the accompanying distance metric that

are optimized with respect to the induced similarity measure. It is remarkable that hand-engineered descriptors perform well under this task *without the need to learn such a measure* for their representations in a supervised manner.

To the best of our knowledge it has never been investigated whether any of the many unsupervised learning algorithms developed over the last couple of years can match this performance without relying on any supervision signals. While we propose an additional benchmark for unsupervised learning algorithms, we do not introduce a new learning algorithm. We rather investigate the performance of the Gaussian RBM (GRBM) [39], its sparse variant (spGRBM) [29] and the mean covariance RBM (mcRBM) [33] without any supervised learning with respect to the matching task. As it turns out, the mcRBM performs comparably to hand-engineered feature descriptors. In fact using a simple heuristic, the mcRBM produces a *compact binary* descriptor that performs better than several state-of-the-art hand-crafted descriptors.

We begin with a brief description of the dataset used for evaluating the matching task, followed by a section on details of the training procedure. In section 4 we present our results, both quantitatively and qualitatively and also mention other models that were tested but not further analyzed because of overall bad performance. Section 5 concludes with a brief summary and an outlook for future work. A review of GRBMs, spGRBMs and mcRBMs is provided in the appendix, section 6, for completeness.

Related work Most similar in spirit to our work are [6, 20, 22]: Like us, [6, 22] are interested in the behavior of unsupervised learning approaches without any supervised steps afterwards. Whereas both investigate high-level representations. [20] learns a compact, binary representation with a very deep autoencoder in order to do fast content-based image search (*semantic hashing*, [36]). Again, these representations are studied with respect to their capabilities to model high-level object concepts. Additionally, various algorithms to learn high-level correspondences have been studied [4, 37, 16] in recent years.

Finding (compact) low-level image descriptors should be an excellent machine learning task: Even hand-designed descriptors have many free parameters that cannot (or should not) be optimized manually. Given ground truth data for correspondences, the performance of supervised learning algorithms is impressive [2]. Very recently, boosted learning with image gradient-based weak learners has shown excellent results [43, 42] on the same dataset used in this paper. See section 2 of [43] for more related work in the space of supervised metric learning.

2 Dataset

At the heart of this paper is a recently introduced dataset for discriminative learning of local image descriptors [2]. It attempts to foster learning optimal low-level image representations using a large and realistic training set of patch correspondences. The dataset is based on more than 1.5 million image patches (64×64 pixels) of three different scenes: the Statue of Liberty (about 450,000 patches), Notre Dame (about 450,000 patches) and Yosemite's Half Dome (about 650,000 patches). The patches are sampled around interest points detected by Difference of Gaussians [27] and are normalized with respect to scale and orientation[1]. As shown in Figure 1, the dataset has a wide variation in lighting conditions, viewpoints, and scales.

The dataset contains also approximately 2.5 million image correspondences. Correspondences between image patches are established via dense surface models obtained from stereo matching (stereo matching, with its epipolar and multi-view constraints, is a much easier problem than unconstrained 2D feature matching). The exact procedure to establish correspondences is more involved and described in detail in [2, Section II]. Because actual 3D correspondences are used, the identified 2D patch correspondences show substantial perspective distortions resulting in a much more realistic dataset than previous approaches [24, 28]. The dataset appears very similar to an earlier benchmark of the same authors [47], yet the correspondences in the novel dataset resemble a much harder problem. The error rate at 95% detection of correct matches for the SIFT descriptor [27] raises from 6% to 26%, the error rate for evaluating patch similarity in pixel space (using normalized sum squared differences) raises from 20% to at least 48% (all numbers are take from [47] and [2] respectively),

[1] A similar dataset of patches centered on multi-scale Harris corners is also available.

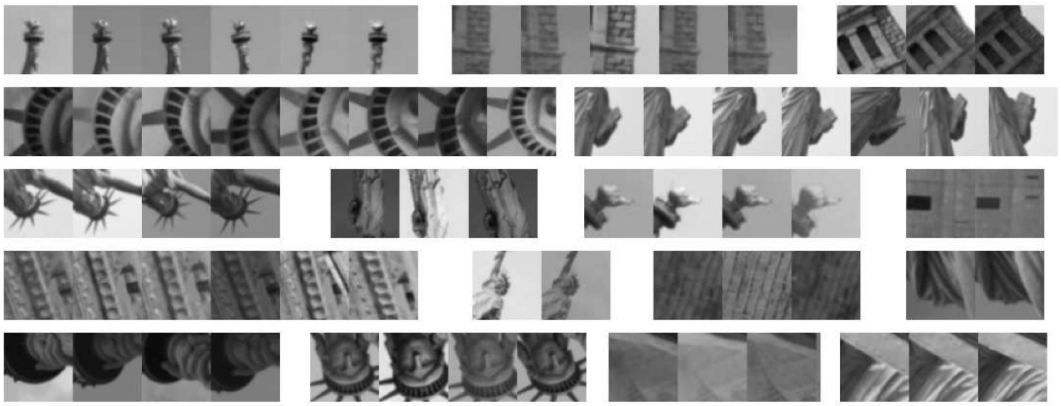

Figure 1: Patch correspondences from the Liberty dataset. Note the wide variation in lighting, viewpoint and level of detail. The patches are centered on interest points but otherwise can be considered random, e.g. there is no reasonable notion of an object boundary possible. Figure taken from [2].

for example. In order to facilitate comparison of various descriptor algorithms a large set of predetermined match/non-match patch pairs is provided. For every scene, sets comprising between 500 and 500,000 pairs (with 50% matching and 50% non-matching pairs) are available.

We don't argue that this dataset subsumes or substitutes any of the previously mentioned benchmarks. Instead, we think that it can serve to complement those. It constitutes an excellent testbed for unsupervised learning algorithms: Experiments considering self-taught learning [32], effects of semi-supervised learning, supervised transfer learning over input distributions with a varying degree of similarity (the scenes of Statue of Liberty and Notredame show architectural structures, while Half Dome resembles a typical natural scenery) and the effect of enhancing the dataset with arbitrary image patches around keypoints can all be conducted in a controlled environment. Furthermore, end-to-end trained systems for (large) classification problems (like [21, 5]) can be evaluated with respect to this type of data distribution and task.

3 Training Setup

Different to [2], our models are trained in an unsupervised fashion on the available patches. We train on one scene (400,000 randomly selected patches from this scene) and evaluate the performance on the test set of every scene. This allows us to investigate the self-taught learning paradigm [32]. We also train on all three scenes jointly (represented by 1.2 million image patches) and then evaluate again every scene individually.

3.1 GRBM/spGRBM

The GRBM and spGRBM (see Appendix, section 6.2) only differ in the setting of the sparsity penalty λ_{sp}, all other settings are the same. We use CD_1 [13] to compute the approximate gradient of the log-likelihood and the recently proposed rmsprop [41] method as gradient ascent method. Compared to standard minibatch gradient ascent, we find that rmsprop is a more efficient method with respect to the training time necessary to learn good representations: it takes at most half of the training time necessary for standard minibatch gradient ascent.

Before learning the parameters, we first scale all image patches to 16×16 pixels. Then we preprocess all training samples by subtracting the vectors' mean and dividing by the standard deviation of its elements. This is a common practice for visual data and corresponds to local brightness and contrast normalization. [39, Section 2.2] gives also a theoretical justification for why this preprocessing step is necessary to learn a reasonable precision matrix $\mathbf{\Lambda}$. We find that this is the only preprocessing scheme that allows GRBM and spGRBM to achieve good results. In addition, it is important to learn

$\boldsymbol{\Lambda}$—setting it to the identity matrix, a common practice [14], also produces dissatisfying error rates. Note that originally it was considered that learning $\boldsymbol{\Lambda}$ is mostly important when one wants to find a good density (i.e. generative) model of the data.

Both GRBM and spGRBM have 512 hidden units. The elements of $\boldsymbol{W}$ are initialized according to $\mathcal{N}(0, 0.1)$, the biases are initialized to 0. rmsprop uses a learning rate of 0.001, the decay factor is 0.9, the minibatch size is 128. We train both models for 10 epochs (this takes about 15 minutes on a consumer GPU for 400000 patches). For the spGRBM we use a sparsity target of $\rho = 0.05$ and a sparsity penalty of $\lambda_{\text{sp}} = 0.2$. spGRBM is very sensitive to settings of λ_{sp} [38]—setting it too high results in dead representations (samples that have no active hidden units) and the results deteriorate drastically.

3.2 mcRBM

mcRBM (see Appendix, section 6.3) training is performed using the code from [33]. We resample the patches to 16×16 pixels. Then the samples are preprocessed by subtracting their mean (patchwise), followed by PCA whitening, which retains 99% of the variance. The overall training procedure (with stochastic gradient descent) is identical to the one described in [33, Section 4]. We train all architectures for a total of 100 epochs, however updating $\boldsymbol{P}$ is only started after epoch 50. We consider two different mcRBM architectures: The first has 256 mean units, 512 factors and 512 covariance units. $\boldsymbol{P}$ is not constrained by any fixed topography. We denote this architecture by *mcRBM*$(256, 512/512)$. The second architecture is concerned with learning more compact representations: It has 64 mean units, 576 factors and 64 covariance units. $\boldsymbol{P}$ is initialized with a two-dimensional topography that takes 5×5 neighborhoods of factors with a stride equal to 3. We denote this model by *mcRBM*$(64, 576/64)$. On a consumer grade GPU it takes 6 hours to train the first architecture on 400000 samples and 4 hours to train the second architecture on the same number of samples.

4 Results

For the results presented in this section (Table 1) we follow the evaluation procedure of [2]: For every scene (Liberty (denoted by LY), Notredame (ND) and Half Dome (HD)), we use the labeled dataset with 100,000 image pairs to assess the quality of a trained model on this scene. In order to save space we do not present ROC curves and only show the results in terms of the *95% error rate* which is the percent of incorrect matches when 95% of the true matches are found: After computing the respective distances for all pairs in a test set, a threshold is determined such that 95% of all matching pairs have a distance below this threshold. Non-matching pairs with a distance below this threshold are considered incorrect matches.

Table 1 consists of two subtables. Table 1a presents the error rates for GRBM, spGRBM and mcRBM when no limitations on the size of representations are placed. Table 1b only considers descriptors that have an overall small memory footprint. For GRBM and spGRBM we use the activations of the hidden units given a preprocessed input patch $\boldsymbol{v}$ as descriptor $D(\boldsymbol{v})$ (see eq. 5, section 6.1):

$$D(\boldsymbol{v}) = \sigma(\boldsymbol{v}^T \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{W} + \boldsymbol{b})$$

For the mcRBM a descriptor is formed by using the activations of the *latent covariance units alone*, see eq. 8, section 6.3:

$$D(\boldsymbol{v}) = \sigma(\boldsymbol{P}^T (\boldsymbol{C}^T \boldsymbol{v})^2 + \boldsymbol{c})$$

This is in accordance with manually designed descriptors. Many of these rely on distributions (i.e. histograms) of intensity gradients or edge directions [27, 28, 1], structural information which is encoded by the covariance units (see also [35, Section 2])[2].

4.1 Distance metrics

As we explicitly refrain from learning a suitable (with respect to the correspondence task) distance metric with a supervised approach, we have to resort to standard distance measures. The Euclidean

[2]Extending the descriptor with mean units degrades results.

Method	Training set	Test set LY	ND	HD
SIFT	–	28.1	20.9	24.7
GRBM $(L_1\ell_1)$	LY	47.6	33.5	41.4
	ND	50.0	33.4	42.5
	HD	49.0	34.0	41.5
	LY/ND/HD	48.7	33.5	42.1
spGRBM $(L_1\ell_1)$	LY	37.9	26.9	34.3
	ND	40.0	28.0	35.4
	HD	39.1	27.9	34.9
	LY/ND/HD	37.5	26.6	33.6
mcRBM $(L_1\ell_2)$	LY	31.3	25.1	34.5
	ND	34.0	25.6	33.0
	HD	31.2	22.3	25.7
	LY/ND/HD	30.8	24.8	33.3
mcRBM (JSD)	LY	34.7	24.2	38.6
	ND	33.3	24.8	44.9
	HD	29.9	22.7	37.6
	LY/ND/HD	30.0	23.1	39.8

(a)

Method	Training set	Test set LY	ND	HD
SIFT	–	31.7	22.8	25.6
BRIEF	–	59.1	54.5	54.9
BRISK	–	79.3	74.8	73.2
SURF	–	54.0	45.5	43.5
BinBoost (8 bytes)	LY	–	16.9	22.8
	ND	20.4	–	18.9
	HD	21.6	14.5	–
ITQ-SIFT (8 bytes)	LY	–	31.1	34.4
	ND	37.0	–	34.3
	HD	37.3	30.5	–
D-Brief (4 bytes)	LY	–	43.1	47.2
	ND	46.2	–	51.3
	HD	53.3	43.9	–
mcRBM (8 bytes)	LY	32.0	35.1	50.5
	ND	41.4	30.1	51.4
	YM	37.9	31.5	47.3
	LY/ND/HD	36.1	27.2	46.1

(b)

Table 1: Error rates, i.e. the percent of incorrect matches when 95% of the true matches are found. All numbers for GRBM, spGRBM and mcRBMs are given within $\pm 0.5\%$. Every subtable, indicated by an entry in the *Method* column, denotes a descriptor algorithm. Descriptor algorithms that do not require learning (denoted by – in the column *Training set*) are represented by one line. The numbers in the columns labeled LY, ND and HD are the error rates of a method on the respective test set for this scene. Supervised algorithms are not evaluated (denoted by –) on the scene they are trained on. The *Training set* LY/ND/HD encompasses 1.2 million patches of all three scenes; this setting is only possible for unsupervised learning methods. (a) Error rates for several unsupervised algorithms without restricting the size of the learned representation. GRBM, spGRBM and mcRBM learn descriptors of dimensionality 512. $(L_1\ell_1)$ denotes that the error rates for a method are with respect to ℓ_1 normalization of the descriptor under the L_1 distance. (b) Results for compact descriptors. BRIEF (32 bytes) [3] and BRISK (64 bytes) [25] are binary descriptors, SURF [1] is a real valued descriptor with 64 dimensions. BinBoost [42], ITQ-SIFT [12] and D-Brief [44] learn compact binary descriptors with supervision. Numbers for BRIEF, BRISK, SURF, BinBoost and ITQ-SIFT are from [42].

distance is widely used when comparing image descriptors. Yet, considering the generative nature of our models we follow the general argumentation of [17] and choose the Manhattan distance, denoted in this text by L_1. We also consider two normalization schemes for patch representations, ℓ_1 and ℓ_2 (i.e. after a feature vector $\boldsymbol{x}$ is computed, its length is normalized such that $\|\boldsymbol{x}\|_1 = 1$ or $\|\boldsymbol{x}\|_2 = 1$).

Given a visible input both (sp)GRBM and mcRBM compute features that resemble parameters of (conditionally) independent Bernoulli random variables. Therefore we consider the Jensen-Shannon divergence (JSD) [26] as an alternative similarity measure. Finally, for binary descriptors, we use the Hamming distance.

4.2 SIFT Baseline

SIFT [27] (both as interest point detector and descriptor) was a landmark for image feature matching. Because of its good performance it is one of the most important basic ingredients for many different kinds of Computer Vision algorithms. It serves as a baseline for evaluating our models. We use vlfeat [45] to compute the SIFT descriptors.

The performance of the SIFT descriptor, ℓ_1-normalized, is reported (using L_1 distance) in Table 1a, first entry. ℓ_1 normalization provides better results than ℓ_2 normalization or no normalization at all. SIFT performs descriptor sampling at a certain scale relative to the Difference of Gaussians peak. In order to achieve good results, it is essential to optimize this scale parameter [2, Figure 6] on every dataset. Table 1b is concerned with evaluating compact descriptors: the first entry shows the performance of SIFT when used as a *128-byte* descriptor (i.e. no normalization applied, but again optimized for the best scale parameter) with L_1 distance.

4.3 Quantitative analysis

Table 1a shows that SIFT performs better than all three unsupervised methods. *mcRBM*(256, 512/512) performs similar to SIFT when trained on Half Dome, albeit at the cost of a 4.5 times larger descriptor representation. The compact *binary* descriptor (the simple binarization scheme is described below) based on *mcRBM*(64, 576/64) performs remarkably well, comparable or even better than several state-of-the-art descriptors (either manually designed or trained in a supervised manner), see Table 1b, last entry. We discuss in more detail several aspects of the results in the following paragraphs.

GRBM and spGRBM spGRBM performs considerably better than its non-sparse version (see Table 1a, second and third entries). This is *not* necessarily expected: Unlike e.g. in classification [8] sparse representations are considered problematic with respect to evaluating distances directly. *Lifetime sparsity* may be after all beneficial in this setting compared to strictly enforced population sparsity. We plan to investigate this issue in more detail in future work by comparing spGRBM to Cardinality restricted boltzman machines [38] on this dataset.

Self-taught paradigm We would expect that the performance of a model trained on the Liberty dataset and evaluated on the Notre Dame scene (and vice versa) should be noticeably better than the performance of a model trained on Half Dome and evaluated on the two architectural datasets. However, this is not what we observe. In particular for the mcRBM (both architectures) it is the opposite: Training on the natural scene data leads to much better performance than the assumed optimal setting.

Jensen-Shannon Divergence Both GRBM and spGRBM perform poorly under the Jensen-Shannon divergence similarity (overall error rates are around 60%), therefore we don't report these numbers in the table. Similar, results for mcRBM under JSD are equally bad. However, if one *scales down* $\boldsymbol{P}$ by a constant (we found the value of 3 appropriate), the results with respect to JSD improve noticeably, see Table 1a, the last entry. The performance on the Half Dome dataset is still not good – the scaling factor should be learned [9], which we also plan for future work.

Compact binary descriptor We were not successful in finding a good compact representation with either GRBM or spGRBM. Finding compact representations for any kind of input data should be done with multiple layers of nonlinearities [20]. But even with only two layers (*mcRBM*$(64, 576/64)$) we learn relatively good compact descriptors. If features are binarized, the representation can be made even more compact (64 bits, i.e. 8 bytes). In order to find a suitable binarization threshold we employ the following simple heuristic: After training on a dataset is finished we histogram *all activations* (values between 0 and 1) *of the training set* and use the *median* of this histogram as the threshold.

4.4 Qualitative analysis

We briefly comment on the developed filters (Figure 2). Unsurprisingly, spGRBM (Figure 2a) and mcRBM (Figure 2b—these are columns from $\boldsymbol{C}$) learn Gabor like filters. At a closer look we make some interesting observations: Figure 2c shows the diagonal elements of $\boldsymbol{\Lambda}^{1/2}$ from a spGRBM. When computing a latent representation, the input $\boldsymbol{v}$ is scaled (elementwise) by this matrix, which, visualized as a 2D image, resembles a Gaussian that is dented at the center, the location of the keypoint of every image patch. The mcRBM also builds filters around the keypoint: Figure 2d shows some unusual filters from $\boldsymbol{C}$. They are centered around the keypoint and bear a strong resemblance to discriminative projections (Figure 2e) that are learned in a supervised way on this dataset [2,

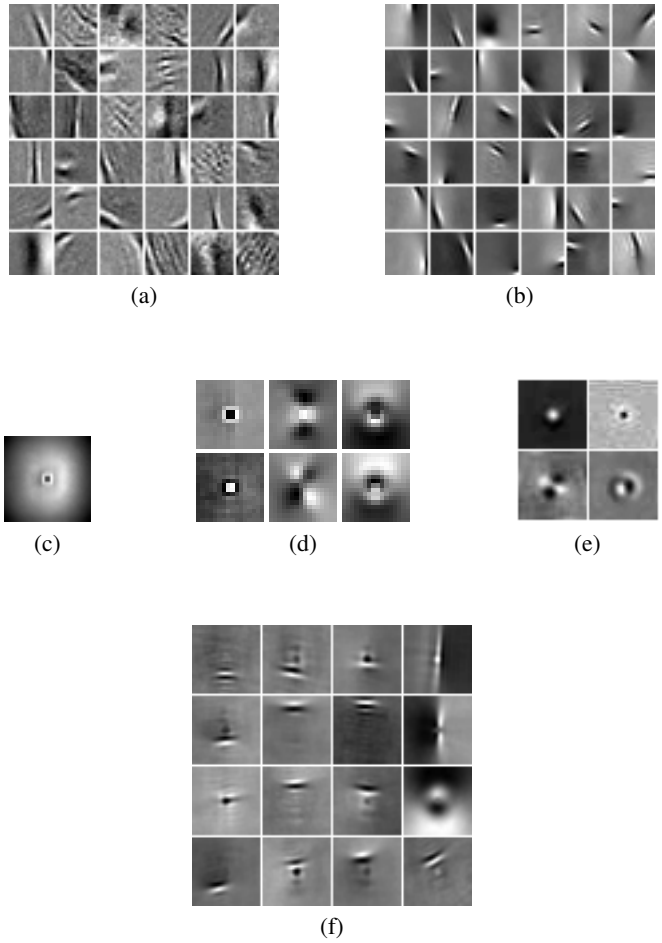

Figure 2: (a) Typical filters learned with spGRBM. (b) Filters from an mcRBM. (c) The pixelwise inverted standard deviations learned with a spGRBM plotted as a 2D image (darker gray intensities resemble lower numerical values). An input patch is elementwise multiplied with this image when computing the latent representation. This figure is generated by training on 32×32 patches for better visibility, but the same qualitative results appear with 16×16 patches. (d) The mcRBM also learns some variants of log-polar filters centered around the DoG keypoint. These are very similar to filters found when optimizing for the correspondence problem in a *supervised* setting. Several of such filters are shown in subfigure (e), taken from [2, Figure 5]. Finally (f), the basic keypoint filters are combined with Garbor filters, if these are placed close to the center; the Garbor filters get systematically arranged around the keypoint filters.

Figure 5]. Qualitatively, the filters in Figure 2d resemble log-polar filters that are used in several state-of-the-art feature designs [28]. The very focused keypoint filters (first column in Figure 2d) are often combined with Gabor filters *placed in the vicinity of the center* – the Garbor filters appear on their own, if they are too far from the center. If an mcRBM is trained with a fixed topography for $\boldsymbol{P}$, one sees that the Gabor filters get systematically arranged around the keypoint (Figure 2f).

4.5 Other models

We also trained several other unsupervised feature learning models: GRBM with nonlinear rectified hidden units[3] [30], various kinds of autoencoders (sparse [7] and denoising [46] autoencoders), K-

[3]Our experiments indicate that rmsprop is in this case also beneficial with respect to the final results: It learns models that perform about 2-3% better than those trained with stochastic gradient descent.

means [7] and two layer models (stacked RBMs, autoencoders with two hidden layers, cRBM [34]). None of these models performed as good as the spGRBM.

5 Conclusion

We start this paper suggesting that unsupervised feature learning should be evaluated (i) without using subsequent supervised algorithms and (ii) more directly with respect to its capacity to find good low-level image descriptors. A recently introduced dataset for discriminatively learning low-level local image descriptors is then proposed as a suitable benchmark for such an evaluation scheme that complements nicely the existing benchmarks. We demonstrate that an mcRBM learns real-valued and binary descriptors that perform comparably or even better to several state-of-the-art methods on this dataset.

In future work we plan to evaluate deeper architectures [20], combined with sparse convolutional features [18] on this dataset. Moreover, ongoing work investigates several algorithms [4, 37] for supervised correspondence learning on the presented dataset.

References

[1] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Proc. ECCV*, 2006.

[2] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *IEEE PAMI*, 2010.

[3] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. Brief: Computing a local binary descriptor very fast. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1281–1298, 2012.

[4] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proc. CVPR*, 2005.

[5] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Proc. CVPR*, 2012.

[6] A. Coates, A. Karpathy, and A. Ng. Emergence of object-selective features in unsupervised feature learning. In *Proc. NIPS*, 2012.

[7] A. Coates, H. Lee, and A. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proc. AISTATS*, 2011.

[8] A. Coates and A. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proc. ICML*, 2011.

[9] G. Dahl, M. Ranzato, A. Mohamed, and G. Hinton. Phone recognition with the mean-covariance restricted boltzmann machine. In *Proc. NIPS*, 2010.

[10] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 2007.

[11] Y. Freund and D. Haussler. Unsupervised learning of distributions on binary vectors using two layer networks. Technical report, University of California, Santa Cruz, 1994.

[12] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *Pattern Analysis and Machine Intelligence*, 2012.

[13] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[14] G. Hinton. A practical guide to training restricted boltzmann machines. Technical report, University of Toronto, 2010.

[15] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[16] G. Huang, M. Mattar, H. Lee, and E. Learned-Miller. Learning to align from scratch. In *Proc. NIPS*, 2012.

[17] Y. Jia and T. Darrell. Heavy-tailed distances for gradient based image descriptors. In *Proc. NIPS*, 2011.

[18] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierarchies for visual recognition. In *Proc. NIPS*, 2010.

[19] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[20] A. Krizhevsky and G. Hinton. Using very deep autoencoders for content-based image retrieval. In *Proc. ESANN*, 2011.

[21] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, 2012.

[22] Q. Le, R. Monga, M. Devin, G. Corrado, K. Chen, M. Ranzato, J. Dean, and A. Ng. Building high-level features using large scale unsupervised learning. In *Proc. ICML*, 2012.

[23] Y. LeCun, F. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proc. CVPR*, 2004.

[24] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *IEEE PAMI*, 28(9):1465–1479, 2006.

[25] S. Leutenegger, M. Chli, and R. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proc. ICCV*, 2011.

[26] J. Lin. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, 1991.

[27] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[28] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE PAMI*, 2005.

[29] V. Nair and G. Hinton. 3-d object recognition with deep belief nets. In *Proc. NIPS*, 2009.

[30] V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. ICML*, 2010.

[31] R. Neal. Probabilistic inference using markov chain monte carlo methods. Technical report, University of Toronto, 1993.

[32] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proc. ICML*, 2007.

[33] M. Ranzato and G. Hinton. Modeling pixel means and covariances using factorized third-order boltzmann machines. In *Proc. CVPR*, 2010.

[34] M. Ranzato, A. Krizhevsky, and G. Hinton. Factored 3-way restricted boltzmann machines for modeling natural images. In *Proc. AISTATS*, 2010.

[35] M. Ranzato, V. Mnih, and G. Hinton. Generating more realistic images using gated mrf's. In *Proc. NIPS*, 2010.

[36] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 2008.

[37] J. Susskind, R. Memisevic, G. Hinton, and M. Pollefeys. Modeling the joint density of two images under a variety of transformations. In *Proc. CVPR*, 2011.

[38] K. Swersky, D. Tarlow, I. Sutskever, R. Salakhutdinov, R. Zemel, and R. Adams. Cardinality restricted boltzmann machines. In *Proc. NIPS*, 2012.

[39] Y. Tang and A. Mohamed. Multiresolution deep belief networks. In *Proc. AISTATS*, 2012.

[40] T. Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proc. ICML*, 2008.

[41] T. Tieleman and G. Hinton. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[42] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit. Boosting binary image descriptors. Technical report, EPFL, 2012.

[43] T. Trzcinski, M. Christoudias, V. Lepetit, and P. Fua. Learning image descriptors with the boosting-trick. In *Proc. NIPS*, 2012.

[44] T. Trzcinski and V. Lepetit. Efficient discriminative projections for compact binary descriptors. In *Proc. ECCV*, 2012.

[45] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the International Conference on Multimedia*, 2010.

[46] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proc. ICML*, 2008.

[47] S. Winder and M. Brown. Learning local image descriptors. In *Proc. CVPR*, 2007.

6 Appendix

6.1 Gaussian-Binary Restricted Boltzmann Machine

The Gaussian-Binary Restricted Boltzmann Machine (GRBM) is an extension of the Binary-Binary RBM [11] that can handle continuous data [15, 39]. It is a bipartite Markov Random Field over a set of visible units, $\boldsymbol{v} \in R^{N_v}$, and a set of hidden units, $\boldsymbol{h} \in \{0,1\}^{N_h}$. Every configuration of units $\boldsymbol{v}$ and units $\boldsymbol{h}$ is associated with an energy $E(\boldsymbol{v}, \boldsymbol{h})$, defined as

$$E(\boldsymbol{v}, \boldsymbol{h}; \theta) = \frac{1}{2}\boldsymbol{v}^T \boldsymbol{\Lambda} \boldsymbol{v} - \boldsymbol{v}^T \boldsymbol{\Lambda} \boldsymbol{a} - \boldsymbol{h}^T \boldsymbol{b} - \boldsymbol{v}^T \boldsymbol{\Lambda} \boldsymbol{W} \boldsymbol{h} \tag{1}$$

with $\theta = (\boldsymbol{W} \in R^{N_v \times N_h}, \boldsymbol{a} \in R^{N_v}, \boldsymbol{b} \in R^{N_h}, \boldsymbol{\Lambda} \in R^{N_v \times N_v})$, the model parameters. $\boldsymbol{W}$ represents the visible-to-hidden symmetric interaction terms, $\boldsymbol{a}$ and $\boldsymbol{b}$ represent the visible and hidden biases respectively and $\boldsymbol{\Lambda}$ is the precision matrix of $\boldsymbol{v}$, taken to be diagonal. $E(\boldsymbol{v}, \boldsymbol{h})$ induces a probability density function over $\boldsymbol{v}$ and $\boldsymbol{h}$:

$$p(\boldsymbol{v}, \boldsymbol{h}; \theta) = \frac{\exp\big(-E(\boldsymbol{v}, \boldsymbol{h}; \theta)\big)}{Z(\theta)} \tag{2}$$

where $Z(\theta)$ is the normalization partition function, $Z(\theta) = \int \sum_{\boldsymbol{h}} \exp\big(-E(\boldsymbol{v}, \boldsymbol{h}; \theta)\big)\, d\boldsymbol{v}$.

Learning the parameters θ is accomplished by gradient ascent in the log-likelihood of θ given N i.i.d. training samples. The log-probability of one training sample is

$$\log p(\boldsymbol{v}) = -\frac{1}{2}\boldsymbol{v}^T \boldsymbol{\Lambda} \boldsymbol{v} + \boldsymbol{v}^T \boldsymbol{\Lambda} \boldsymbol{a} + \sum_j^{N_h} \log\left(1 + \exp\left(\sum_i^{N_v} \boldsymbol{v}_i^T (\boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{W})_{ij} + \boldsymbol{b}_j\right)\right) - Z(\theta) \tag{3}$$

Evaluating $Z(\theta)$ is intractable, therefore algorithms like Contrastive Divergence (CD) [13] or persistent CD (PCD) [40] are used to compute an approximation of the log-likelihood gradient. The bipartite nature of an (G)RBM is an important aspect when using these algorithms: The visible units are conditionally independent given the hidden units. They are distributed according to a diagonal Gaussian:

$$p(\boldsymbol{v} \mid \boldsymbol{h}) \sim \mathcal{N}(\boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{W} \boldsymbol{h} + \boldsymbol{a}, \boldsymbol{\Lambda}^{-1}) \tag{4}$$

Similarly, the hidden units are conditionally independent given the visible units. The conditional distribution can be written compactly as

$$p(\boldsymbol{h} \mid \boldsymbol{v}) = \sigma(\boldsymbol{v}^T \boldsymbol{\Lambda}^{\frac{1}{2}} \boldsymbol{W} + \boldsymbol{b}) \tag{5}$$

where σ denotes the element-wise logistic sigmoid function, $\sigma(z) = 1/(1 + e^{-z})$.

6.2 Sparse GRBM

In many tasks it is beneficial to have features that are only rarely active [29, 8]. Sparse activation of a binary hidden unit can be achieved by specifying a sparsity target ρ and adding an additional penalty term to the log-likelihood objective that encourages the actual probability of unit j of being active, q_j, to be close to ρ [29, 14]. This penalty is proportional to the negative KL divergence between the hidden unit marginal $q_j = \frac{1}{N}\sum_n p(\boldsymbol{h}_j = 1 \mid \boldsymbol{v}_n)$ and the target sparsity:

$$\lambda_{\text{sp}}\big(\rho \log q_j + (1-\rho)\log(1-q_j)\big), \tag{6}$$

where λ_{sp} represents the strength of the penalty. This term enforces sparsity of feature j *over* the training set, also referred to as *lifetime sparsity*. The hope is that the features for one training sample are then encoded by a sparse vector, corresponding to *population sparsity*. We denote a GRBM with a sparsity penalty $\lambda_{\text{sp}} > 0$ as *spGRBM*.

6.3 Mean-Covariance Restricted Boltzmann Machine

In order to model pairwise dependencies of visible units gated by hidden units, a third-order RBM can be defined with a weight w_{ijk} for each triplet v_i, v_j, h_k. By factorizing and tying these weights, parameters can be reduced to a filter matrix $\boldsymbol{C} \in R^{N_v \times F}$ and a *pooling* matrix $\boldsymbol{P} \in R^{F \times N_h}$. $\boldsymbol{C}$ connects the input to a set of *factors* and $\boldsymbol{P}$ maps factors to hidden variables. The energy function for this *cRBM* [34] is

$$E_c(\boldsymbol{v}, \boldsymbol{h}_c; \theta) = -(\boldsymbol{v}^T \boldsymbol{C}^T)^2 \boldsymbol{P} \boldsymbol{h}_c - \boldsymbol{c}^T \boldsymbol{h}_c \tag{7}$$

where $(\cdot)^2$ denotes the element-wise square operation and $\theta = \{\boldsymbol{C}, \boldsymbol{P}, \boldsymbol{c}\}$. Note that $\boldsymbol{P}$ has to be non-positive [34, Section 5]. The hidden units of the cRBM are still conditionally independent given the visible units, so inference remains simple. Their conditional distribution (given visible state $\boldsymbol{v}$) is

$$p(\boldsymbol{h}_c \mid \boldsymbol{v}) = \sigma(\boldsymbol{P}^T (\boldsymbol{C}^T \boldsymbol{v})^2 + \boldsymbol{c}) \tag{8}$$

The visible units are coupled in a Markov Random Field determined by the setting of the hidden units:

$$p(\boldsymbol{v} \mid \boldsymbol{h}_c) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}) \tag{9}$$

with

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{C} \mathrm{diag}(-\boldsymbol{P} \boldsymbol{h}_c) \boldsymbol{C}^T \tag{10}$$

As equation 9 shows, the cRBM can only model Gaussian inputs with zero mean. For general Gaussian-distributed inputs the cRBM and the GRBM can be combined into the *mean-covariance RBM* (mcRBM) by simply adding their respective energy functions:

$$E_{mc}(\boldsymbol{v}, \boldsymbol{h}_m, \boldsymbol{h}_c; \theta, \theta') = E_m(\boldsymbol{v}, \boldsymbol{h}_m; \theta) + E_c(\boldsymbol{v}, \boldsymbol{h}_c, \theta') \tag{11}$$

$E_m(\boldsymbol{v}, \boldsymbol{h}_m; \theta)$ denotes the energy function of the GRBM (see eq. 1) with $\boldsymbol{\Lambda}$ fixed to the identity matrix. The resulting conditional distribution over the visible units, given the two sets of hidden units $\boldsymbol{h}_m$ (*mean* units) and $\boldsymbol{h}_c$ (*covariance* units) is

$$p(\boldsymbol{v} \mid \boldsymbol{h}_m, \boldsymbol{h}_c) \sim \mathcal{N}(\boldsymbol{\Sigma} \boldsymbol{W} \boldsymbol{h}_m, \boldsymbol{\Sigma}) \tag{12}$$

with $\boldsymbol{\Sigma}$ defined as in eq. 10. The conditional distributions $p(\boldsymbol{h}_m|\boldsymbol{v})$ and $p(\boldsymbol{h}_c|\boldsymbol{v})$ are still as in eq. 5 and eq. 7 respectively. The parameters θ, θ' can again be learned using approximate Maximum Likelihood Estimation, e.g. via CD or PCD. These methods require to sample from $p(\boldsymbol{v}|\boldsymbol{h}_m, \boldsymbol{h}_c)$, which involves an expensive matrix inversion (see eq. 10). Instead, samples are obtained by using Hybrid Monte Carlo (HMC) [31] on the mcRBM free energy [33].